27. (amended) A computer program product in a computer readable medium for monitoring a plurality of related threads, the computer program product comprising:

first instructions for polling the plurality of related threads for status information;

second instructions for responsive to receiving the status information, determining whether a thread within a plurality of related threads is inactive; and

third instructions for responsive to a determination that a thread within the plurality of related threads is inactive, initiating cleanup processes for the thread based on the status information.

A7

28. (amended) A computer program product in a computer readable medium for monitoring a plurality of related threads, the computer program product comprising:

first instructions for polling the plurality of related threads for status information;

second instructions, responsive to receiving the status information, for determining whether a thread within a plurality of related threads is inactive; and

third instructions, responsive to an occurrence of inactivity in a thread within the plurality of related threads in which the inactivity is due to an event, for initiating cleanup processes based on the status information.

REMARKS

Claims 1-28 are pending in the present application. Claims 1, 4, 11-12, 14, 17, 24-25, and 27-28 were amended. Reconsideration of the claims is respectfully requested.

In addition, Applicants would like to thank the Examiner for his courtesy in allowing Applicants a telephone interview on November 7, 2002. In this interview, Applicants discussed the term "status information" as being defined in the specification. Support in the present invention for the term "status information" is cited in the discussion below.

I. 35 U.S.C. § 112, First Paragraph

The Examiner has objected to the specification under 35 U.S.C. § 112, first paragraph, as failing to adequately teach how to make and/or use the invention in claims

1, 2, 4, 11, 14, 15, 17, 24, 27, and 28. Additionally, the Examiner rejected the claims under the same reasons. This rejection is respectfully traversed.

With regard to claims 1, 2, 4, 11, 14, 15, 17, 24, 27, and 28, the Examiner stated:

> Referring to claims 1, 2, 11, 14, 15, 24, 27, and 28, "status information" needs to be defined in the specification because it is not explicitly understood what status-type of information it constitutes of.

Nothing more than objective enablement is required under 35 U.S.C. § 112, first paragraph, and therefore it is irrelevant whether the requisite teaching is made through broad terminology or illustrative examples. *In re Wright*, 999 F.2d 1557, 1561, 27 U.S.P.Q.2d 1510, 1513 (Fed. Cir. 1993); *In re Marzocchi*, 439 F.2d 220, 223, 169 U.S.P.Q. 367, 369 (CCPA 1971). In the present application, "status information" is demonstrated through example. The specification, for example, mentions using the print job status entry as possible status information at page 19, line 27 to page 20, line 14.

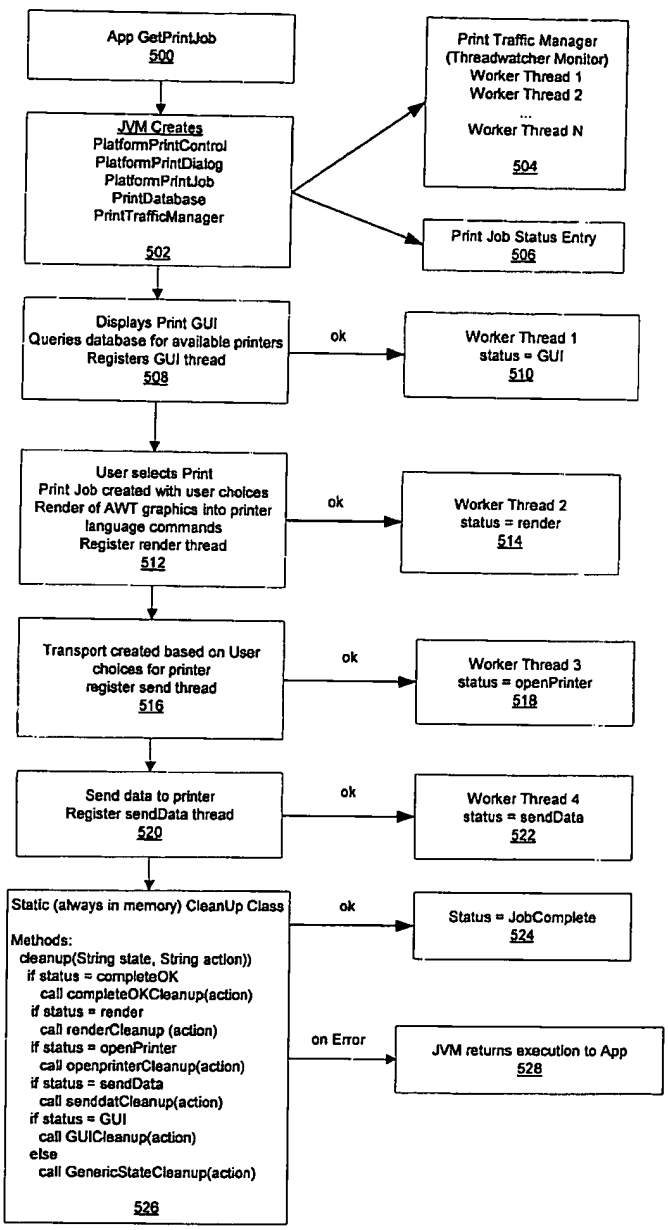
> In **Figure 5**, the thread watcher monitor monitors the different phases of the print job and records status information as the print job progresses. In block **506**, a print job status entry is created to store the status of various print jobs being processed by the worker threads. In block **508**, a print GUI is displayed to user, which queries the user for available printer. In this example, worker thread 1 is registered to the GUI in the print traffic manager. This status is seen in block **510** in which the status for worker thread 1 is set equal to GUI. Entries within PrintJobStatusEntry are used to update changes in status for various threads. Potential problems during this portion of the print job include for example, no printers are installed or that the user is not allowed to print. If a problem such as that occurs and worker thread 1 becomes inactive, cleanup may be performed through thread using the status information recorded in print job status entry.

The specification, by example, also demonstrates at page 21, line 27 to page 22, line 8, that the status information determines which cleanup process to be used on the thread:

> In the depicted example, the type of cleanup that occurs in block **526** depends on the status stored for the particular thread. The following are examples of different methods used in cleanup based on the status for a thread: for status equal to completeOK, a call is made to a method completeOKCleanup; for a status equal to render, a call is made to a method renderCleanup; for a status equal to openPrinter, a call is made to a method openprinterCleanup; for a status equal to sendData, a call is made to a method sendataCleanup; and for a status equal to GUI, a call is made to a method GUICleanup. In this example, if the status is not one of the ones listed above, a call is made to a method GenericStateCleanup.

Furthermore, in Figure 5, for example, blocks **510**, **514**, **518**, **522**, **524**, and **526** show various types of status information examples stored for a particular thread. Figure 5 is reproduced below.

Figure 5
AT9-99-149



In view of the above, the specification, as originally filed, provides sufficient support and description to enable one of ordinary skill in the art to make and use the invention as recited in claims 1, 2, 11, 14, 15, 24, 27, and 28.

With regard to dependent claims 4 and 17, these claims have been amended to remove the word "first" from the phrase "first class" in response to the Examiner's comments.

Therefore, in view of the above, Applicants submit that the features of the pending claims are completely supported by the specification and the specification provides a description of the claimed subject matter that reasonably conveys to those of ordinary skill in the relevant art that the inventors had possession of the claimed invention at the time the application was filed. Thus, Applicants respectfully request that the rejections of the specification under 35 U.S.C. §112, first paragraph be withdrawn.

II. 35 U.S.C. § 112, Second Paragraph

The Examiner has rejected claims 1, 2, 4, 11, 12, 14, 15, 17, 24, 25, 27, and 28 under 35 U.S.C. §112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter, which applicants regard as the invention. This rejection is respectfully traversed. With regard to claims 1, 2, 4, 11, 12, 14, 15, 17, 24, 25, 27, and 28, the Examiner stated:

> Referring to claims 1, 2, 11, 14, 15, 24, 27, and 28, the term "status information" is indefinite. For example, the 2nd step determination is made to determine whether a thread is active (based on status information). Then in the last step (lines 12-15), we disregard that determination and initiate cleanup anyway. These steps (1-2) are not related to the method of the invention.

The term "status information" is not indefinite as explained above in Applicants' response to the rejection under 35 U.S.C. §112, first paragraph. The term "status information" is supported in the specification. Furthermore, the Examiner's statement that we disregard the determination whether a thread is active in the second step and initiate cleanup anyway is incorrect. The determination whether the thread is active in the second step is needed to carry out the third step in the claim. For example, the third step in claim 1 recites "responsive to an absence of a determination that a thread within

the plurality of related threads is active, initiating cleanup processes for the thread based on the status information." Thus, the absence of a determination that the thread is active in the second step results in the initiation of a cleanup process in the third step based upon the status information. However, Applicants have amended claims 1, 11, 14, 24, 27, and 28 to make the subject matter even clearer.

The Examiner rejected claims 4 and 17 for including the word "first" in front of "class." Applicants have amended claims 4 and 17 to recite simply "a class," thus obviating the rejection.

The Examiner rejected claims 12 and 25 for reciting "wherein the event is a period of time." While not necessarily agreeing that the Examiner's rejection has merit, Applicants have amended claims 12 and 25 to add "an occurrence of," so that the phrase now reads, "wherein the event is an occurrence of a period of time."

Applicants respectfully submit that claims 1, 2, 4, 11, 12, 14, 15, 17, 24, 25, 27, and 28 meet the requirements of 35 U.S.C. §112, second paragraph and respectfully request that the rejection of these claims under 35 U.S.C. §112, second paragraph be withdrawn.

III. 35 U.S.C. § 103, Obviousness

The examiner has rejected claims 1-3, 6, 11, 13-16, 19, 24, and 26-28 under 35 U.S.C. § 103(a) as being unpatentable over Yeager (US 6,418,542). This rejection is respectfully traversed.

A. The Examiner bears the burden of establishing a *prima facie* case of obviousness.

The examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Applicants believe the Examiner has failed to meet this burden for the following reasons:

B. The prior art does not teach the problem or its source.

The present invention recognizes the problem of determining whether a thread within a multi-threaded process is inactive, and if so, cleaning up that inactive thread's

resources to enable a more efficient use of the system's resources. *Yeager* does not teach the problem or its source. Instead, *Yeager* is directed towards a system that allows threads in a multi-threaded process to continue executing when a single thread within the process receives a critical signal and crashes. Results of a thread crashing typically causes the entire operating system to shut down, or if the thread crashes on a network server, brings down the entire network. *Yeager* teaches handling critical signals directed to a thread in the process and not having the entire process, which has other threads running in it, terminate. One of ordinary skill in the art would therefore not be motivated to modify the reference in the manner required to form the solution disclosed in the claimed invention.

C. References must teach or suggest all elements of the rejected claims

With regard to independent claims 1, 11, 14, 24, 27, and 28, the Examiner stated:

> Referring to claim 1, 11, 14, 24, 27, and 28, Yeager discloses a data processing system for monitoring a plurality of related threads with the following steps:
>
> **polling** the **plurality** or related **threads** for **status information** ("**polling thread**") for "**input events**", col. 2, lines 54-59 and "**plurality of threads**", col. 3 lines 35-41);
>
> determining whether a **thread** within a **plurality of related threads** is **active** as a response to **receiving status information** ("**multi-threaded process**" contains "**information** on each **thread** in the **plurality of threads,**" col. 3, lines 37-39, and "**active**", "**threads**", "**critical signal**" determines if active, col. 3, lines 16-22);
>
> initiating **cleanup** processes for the **thread** based on the **status information** ("**cleaning** up a particular **thread's resources**:, "**critical signal handler**" decides when to clean based on its access to the **"data in shared memory**", col. 8, lines 38-46).
>
> Yeager fails to explicitly teach:
>
> performing the cleanup process in response to an absence of determination that a thread within the plurality of related threads is active.
>
> However, it would have been obvious to one of ordinary skill in the art at the time the invention was made to clean up errors whenever they occur, or more specifically, perform a "cleanup" when the system fails to determine if the threads are active. This would increase the efficiency because in the instance of being active, the unnecessary resource could be used by other threads.

(*Office Action*, dated August 27, 2002, pages 4-5).

For an invention to be prima facie obvious, the prior art must teach or suggest all claim limitations. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). Amended independent claims 1, 11, 14, 24, 27, and 28 recite the limitations of polling a thread for status information, determining whether a thread is inactive in response to receiving the status information, and responsive to a determination that the thread is inactive, initiating cleanup processes for the thread based on the status information. Amended independent claim 1, which is representative of amended independent claims 11, 14, 24, 27, and 28, is reproduced below:

> 1. A method in a data processing system for monitoring a plurality of related threads, the method comprising the data processing system implemented steps of:
>
> polling the plurality of related threads for status information;
>
> responsive to receiving the status information, determining whether a thread within a plurality of related threads is inactive; and
>
> responsive to a determination that a thread within the plurality of related threads is inactive, initiating cleanup processes for the thread based on the status information.

The claimed invention teaches polling a thread for status information. The Examiner states that *Yeager* teaches this feature in the following passages:

> In yet another embodiment an input polling thread contained within the process is instructed to discontinue polling for input events directed to the offending thread. In yet another embodiment a core file of the process is made at the time it receives the critical signal even though the entire process is not shut down.

(*Yeager*, col. 2, ll. 54-59).

> In yet another embodiment the system includes a memory shared by multi-threaded process in the system which contains information on each thread in the plurality of threads. In yet another embodiment the system includes an input polling thread in the process which is instructed by the critical signal thread to discontinue polling for input events directed to the offending thread.

(*Yeager*, col. 3, ll. 35-41).

These passages from *Yeager* do not teach or suggest polling a thread for status information. Instead, *Yeager* teaches handling an input event directed to a thread within a process operating in a multi-threaded system. The process is alerted that an input event

affecting one of its active connection threads has been received. A special thread referred to as an input polling thread is enabled and is used to determine which of the threads in the process has an event directed to it. That thread is triggered to handle the input event. However, *Yeager* fails to teach or suggest polling a thread for status information as in the claimed invention, because *Yeager* is directed toward polling for input events (events directed to a thread):

> At step 314, an input polling thread is instructed to no longer poll on the offending thread. The input polling thread is described in greater detail in co-pending application Ser. No. 09/067,546, entitled "METHOD AND APPARATUS FOR DETECTING INPUT DIRECTED TO A THREAD IN A MULTI-THREADED PROCESS," which is incorporated herein by reference. In the described embodiment, a process has an input polling thread that detects and routes input events directed to the process to the appropriate thread in the process and eliminates the need for each thread in the process to be actively looking for input events directed to it.

(*Yeager,* col. 8, ll. 55-66). Even though *Yeager* teaches polling a thread, *Yeager* does not teach or suggest polling a thread *for status information* as in the claimed invention. Instead, *Yeager* teaches polling a thread *for input events*. "Input events," as used in *Yeager*, differs from "status information," as used in the present invention. Input events are generally some type of activity generated by a user or some external source, such as another computer network, which are directed to a thread. Polling of the input events involves detecting and routing those input events to the appropriate thread. "Status information," on the other hand, is merely the status of the thread.

In addition, an input event may itself affect the status information of a thread. An input event can cause the thread's status information to change in response to the event received. Thus, for the reasons set forth above, the polling process in *Yeager* fails to teach or suggest to one of ordinary skill in the art the step of polling a thread for status information as claimed in the present invention.

Furthermore, the claimed invention teaches determining whether a thread is inactive in response to receiving status information. The Examiner refers to the following passages as evidence that *Yeager* teaches the feature of determining whether a thread is active:

> In another aspect of the present invention a computer system having a multi-threaded process capable of executing active connection threads

where the system is arranged such that when a critical signal is generated for an offending thread, other threads continue within the process is described.

(*Yeager*, col. 3, ll. 16-22).

> In yet another embodiment the system includes a memory shared by multi-threaded process in the system which contains information on each thread in the plurality of threads. In yet another embodiment the system includes an input polling thread in the process which is instructed by the critical signal thread to discontinue polling for input events directed to the offending thread.

(*Yeager*, col. 3, ll. 34-41).

Nothing is present in these cited passages that teaches or suggests determining whether a thread is inactive in response to receiving status information. The passages above refer to allowing non-offending threads in a multi-threaded process to continue to function despite the presence of an offending thread. However, even assuming arguendo that the Examiner is correct in asserting that *Yeager* discloses that a critical signal determines the activity of a thread ("critical signal determines if active" from column 3, line 16-22), this interpretation still does not demonstrate that *Yeager* teaches or suggests the present invention. Even if *Yeager* determines if a thread is active or inactive, the reference does not teach the feature of determining whether a thread is inactive *in response to receiving status information*. Therefore, even if we assume, for the purpose of argument, that *Yeager* teaches that a critical signal can determine if a thread is active or inactive, *Yeager* still fails to provide any teaching or suggestion that the determination that a thread is inactive is performed in response to receiving status information.

Moreover, the claimed invention initiates a cleanup process for a thread based on the status information and in response to a determination that a thread is not active. *Yeager* fails to teach or suggest this step. Although *Yeager* performs a cleanup process on a thread, *Yeager* does not teach or suggest performing the cleanup process in response to a determination that the thread is inactive. Figure 2 illustrates the *Yeager* cleanup process:

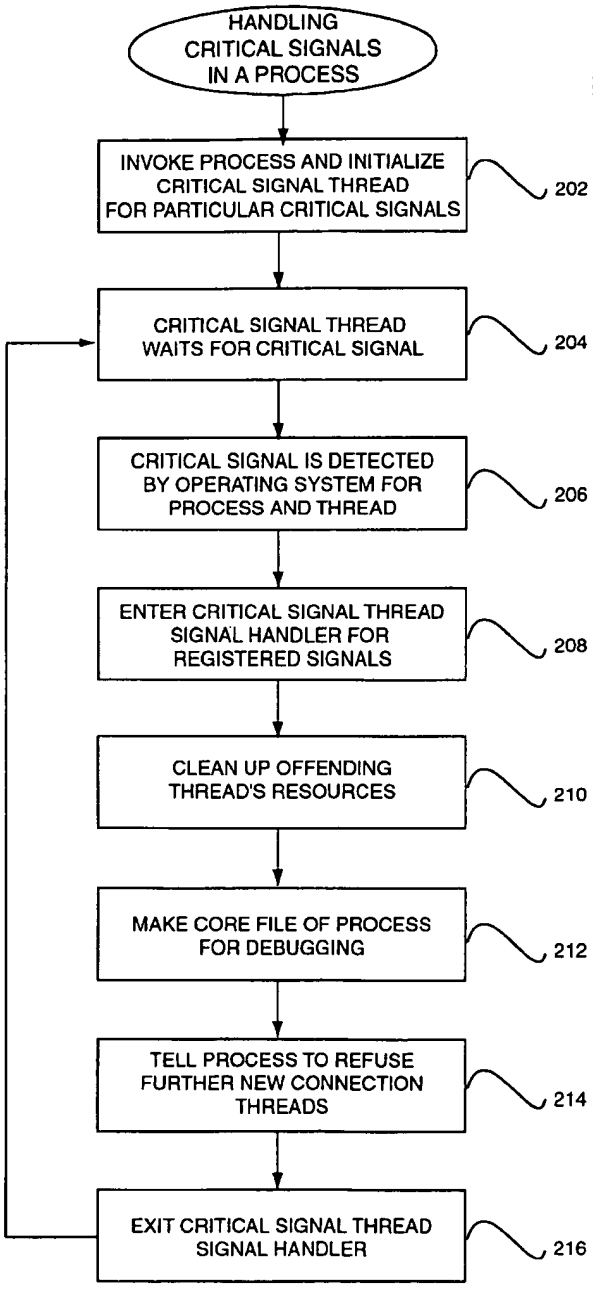

HANDLING CRITICAL SIGNALS IN A PROCESS
INVOKE PROCESS AND INITIALIZE CRITICAL SIGNAL THREAD FOR PARTICULAR CRITICAL SIGNALS
202
CRITICAL SIGNAL THREAD WAITS FOR CRITICAL SIGNAL
204
CRITICAL SIGNAL IS DETECTED BY OPERATING SYSTEM FOR PROCESS AND THREAD
206
ENTER CRITICAL SIGNAL THREAD SIGNAL HANDLER FOR REGISTERED SIGNALS
208
CLEAN UP OFFENDING THREAD'S RESOURCES
210
MAKE CORE FILE OF PROCESS FOR DEBUGGING
212
TELL PROCESS TO REFUSE FURTHER NEW CONNECTION THREADS
214
EXIT CRITICAL SIGNAL THREAD SIGNAL HANDLER
216

FIG. 2

> FIG. 2 is a flowchart showing a method of terminating an offending thread in a process without terminating the entire process in accordance with one embodiment of the present invention. At step 202 a process is executed and a critical signal thread associated with the process is initialized. When the process is started, it registers certain signals that it will catch and process. These signals include normal processing signals and critical signals that indicate an interrupt. For example, in the Solaris Operating System from Sun Microsystems, signals 10 and 11 are critical signals or interrupts indicating an illegal instruction and illegal memory access. When a signal is registered by a process at start time, the critical signal thread is enabled to handle that signal. Thus, at process start-up, the critical signal thread is enabled to catch any critical signals registered by the process. With previous systems, the process ignored critical signals and caught only normal or non-disruptive signals. When the signal is ignored, the operating system terminates the process. In the described embodiment, the signals are registered by the process and the critical signal thread is notified to catch those signals.

(*Yeager,* col. 6, ll. 37-57).

> The critical signal thread has several function calls it can make during its execution. In the described embodiment, one such function is known as the critical signal handler. The signal handler is a portion of code of the critical signal thread that handles the incoming registered signal. In other preferred embodiments, the critical signal thread can contain more than one critical signal handler for handling different types of critical signals. As mentioned above, in the Solaris Operating System signals 10 and 11 are critical signals that are handled by the same signal handler. In other preferred embodiments, there may be more than one signal handler for handler such signals. At step 208, the critical signal handler is invoked. When a signal is received by the crash thread from a signal queue maintained by the operating system, the thread calls the appropriate registered signal handler for handling the incoming critical signal.

(*Yeager,* col. 7, ll. 5-20).

> Once the signal handler function has been called, the thread begins cleaning up the offending thread's resources. In the described embodiment, this is done by calling another function of the crash thread referred to as thread clean-up. This is done at step 210. The critical signal thread has access to shared memory and can determine which functions need to be called. The process of cleaning up an offending thread's resources is discussed in greater detail in FIG. 3.

(*Yeager,* col. 7, ll. 26-33). As can be seen from Figure 2, *Yeager's* cleanup process is performed in response to receiving a critical signal caused by an offending thread. A critical signal indicates that the thread is not operating properly, such as an illegal

instruction or illegal memory access. There is no teaching or suggestion in *Yeager* to initiate cleanup of a thread's resources simply in response to a determination that the thread is inactive as in the claimed invention.

Similarly, *Yeager* fails to teach or suggest performing a cleanup process based on the status information as recited in the claimed invention. The Examiner cites the following as evidence that *Yeager* teaches this feature:

> At step 308 the critical signal handler determines which function calls to make in order to clean up the thread's remaining resources. Since the critical signal thread has access to all the data in shared memory, it can determine which function calls are necessary for cleaning up a particular thread's resources. At step 310 the critical signal handler causes the system to remove all references to the thread from the shared allocated memory 116, as shown in FIG. 1.

(*Yeager,* col. 8, ll. 38-46). The cited passage demonstrates that *Yeager* uses a critical signal handler to make function calls to cleanup an offending thread's resources. However, the passage makes no mention of basing the cleanup process on the status information. Thus, *Yeager* teaches initiating a cleanup process based on the receipt of a critical signal caused by an offending thread, rather than initiating a cleanup process based on the status information as claimed in the present invention.

D. Stating that it is obvious to try or make a modification or combination without a suggestion in the prior art is not *prima facie* obviousness

The mere fact that a prior art reference can be readily modified does not make the modification obvious unless the prior art suggested the desirability of the modification. *In re Laskowski*, 871 F.2d 115, 10 U.S.P.Q.2d 1397 (Fed. Cir. 1989) and also *see In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992) and *In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1993). The Examiner may not merely state that the modification would have been obvious to one of ordinary skill in the art without pointing out in the prior art a suggestion of the desirability of the proposed modification. The claimed invention includes the feature of performing a cleanup process in response to a determination that a thread within the plurality of threads is inactive. In contrast, *Yeager* teaches performing a cleanup process on a particular thread's resources, but does not teach or suggest initiating a cleanup process for an inactive thread. In fact, the Examiner states that *Yeager* explicitly fails to teach "performing the cleanup process in response to

an absence of a determination that a thread within the plurality of related threads is active." (*Office Action,* page 5). The rationale given for the modification is not based on the prior art. Instead, the Examiner states his personal opinion using hindsight and "increased efficiency" as the rationale. The fact that a thread is inactive does not mean that one would perform a cleanup process on it. For example, the thread may be alive but suspended and inactive. One would not necessarily desire to perform a cleanup process on that inactive thread. Thus, since *Yeager* does not teach performing the cleanup process in response to a determination that a thread is inactive, nor does the Examiner point to any reference that suggests this feature be combined with the contents of the *Yeager* reference, Applicants respectfully submit that *Yeager* cannot be modified to produce the claimed invention.

E. The prior art reference actually teaches away from the claimed invention

"It is impermissible within the framework of section 103 to pick and choose from any one reference only so much of it as will support a given position, to the exclusion of other parts necessary to the full appreciation of what such reference fairly suggests to one of ordinary skill in the art." *In re Hedges*, 228 U.S.P.Q. at 687. *Yeager* does not teach, suggest, or give any incentive to make the needed changes to reach the presently claimed invention. *Yeager* actually teaches away from the presently claimed invention because it performs its cleanup process based on the receipt of a critical signal caused by an offending thread, rather than performing its cleanup process in response to a determination that the thread is inactive. Absent the Examiner pointing out some teaching or incentive to implement *Yeager* to perform a cleanup process on a thread in response to on a determination that a thread is inactive, one of ordinary skill in the art would not be led to modify *Yeager* to reach the present invention when the reference is examined as a whole.

Furthermore, *Yeager* teaches away from the claimed invention since *Yeager* teaches initiating a cleanup process based on the receipt of a critical signal caused by an offending thread, rather than initiating a cleanup process based on the status information obtained. Thus, one of ordinary skill in the art would not be motivated from the reference to make the changes necessary to derive the present invention from the reference's teachings.

F. The presently claimed invention may be reached only through improper hindsight

In addition, the Examiner may not make modifications to the prior art using the claimed invention as a model for the modifications. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780, 1783-84 (Fed. Cir. 1992). "The mere fact that the prior art may be modified in the manner suggested by the Examiner does not make the modification obvious unless the prior art has suggested the desirability of the combination." *Id.* In other words, unless some teaching exists in the prior art for the suggested modification, merely asserting that such a modification would be obvious to one of ordinary skill in the art is improper and cannot be used to meet the burden of establishing a *prima facie* case of obviousness. Such reliance is an impermissible use of hindsight with the benefit of Applicants' disclosure.

Absent some teaching, suggestion, or incentive in the prior art, *Yeager* cannot be properly modified to form the claimed invention; the presently claimed invention can be reached only through an impermissible use of hindsight with the benefit of Applicants' invention as a model.

Therefore, the rejection of claims 1, 11, 14, 24, 27, and 28 under 35 U.S.C. § 103 has been overcome.

VI. Dependent Claims

If an independent claim is nonobvious under 35 U.S.C. §103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). Claims 2-10, 12-13, 15-23, and 25-26 are dependent claims that depend on independent claims 1, 11, 14, 24, 27, and 28. Applicants have already demonstrated claims 1, 11, 14, 24, 27, and 28 to be in condition for allowance. Applicants respectfully submit that claims 2-10, 12-13, 15-23, and 25-26 are also allowable, at least by virtue of their dependency on allowable claims.

Therefore, the rejection of claims 1-28 under 35 U.S.C. § 103 has been overcome.

VII. Conclusion

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance.

The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: 11/11/02

Respectfully submitted,

Cathrine K. Kinslow
Reg. No. 51,886
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
Agent for Applicants

IN THE CLAIMS:

A marked-up version of the amended claims is as follows:

1. (amended) A method in a data processing system for monitoring a plurality of related threads, the method comprising the data processing system implemented steps of:

polling the plurality of related threads for status information;

responsive to receiving the status information, determining whether a thread within a plurality of related threads is inactive [active]; and

responsive to [an absence of] a determination that a thread within the plurality of related threads is inactive [active], initiating cleanup processes for the thread based on the status information.

4. (amended) The method of claim 1, wherein the single thread is part of a [first] class.

11. (amended) A method in a data processing system for monitoring a plurality of related threads, the method comprising the data processing system implemented steps of:

polling the plurality of related threads for status information;

responsive to receiving the status information, determining whether [an error has occurred in] a thread within a plurality of related threads is inactive [active]; and

responsive to an occurrence of inactivity in a thread within the plurality of related threads in which the inactivity is due to an event, initiating cleanup processes based on the status information.

12. (amended) The method of claim 11, wherein the event is an occurrence of a period of time.

14. (amended) A data processing system for monitoring a plurality of related threads, the data processing system comprising:

polling means for polling the plurality of related threads for status information;

determining means, responsive to receiving the status information, for determining whether a thread within a plurality of related threads is inactive [active]; and

initiating means, responsive to [an absence of] a determination that a thread within the plurality of related threads is inactive [active], for initiating cleanup processes for the thread based on the status information.

17. (amended) The data processing system of claim 14, wherein the single thread is part of a [first] class.

24. (amended) A data processing system for monitoring a plurality of related threads, the data processing system comprising:

polling means for polling the plurality of related threads for status information;

determining means, responsive to receiving the status information, for determining whether [an error has occurred in] a thread within a plurality of related threads is inactive [active]; and

initiating means, responsive to an occurrence of inactivity in a thread within the plurality of related threads in which the inactivity is due to an event, for initiating cleanup processes based on the status information.

25. (amended) The data processing system of claim 24, wherein the event is an occurrence of a period of time.

27. (amended) A computer program product in a computer readable medium for monitoring a plurality of related threads, the computer program product comprising:

first instructions for polling the plurality of related threads for status information;

second instructions for responsive to receiving the status information, determining whether a thread within a plurality of related threads is inactive [active]; and

third instructions for responsive to [an absence of] a determination that a thread within the plurality of related threads is inactive [active], initiating cleanup processes for the thread based on the status information.

28. (amended) A computer program product in a computer readable medium for monitoring a plurality of related threads, the computer program product comprising:

first instructions for polling the plurality of related threads for status information;

second instructions, responsive to receiving the status information, for determining whether [an error has occurred in] a thread within a plurality of related threads is inactive [active]; and

third instructions, responsive to an occurrence of inactivity in a thread within the plurality of related threads in which the inactivity is due to an event, for initiating cleanup processes based on the status information.